In [36]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [37]:

```python
data=pd.read_csv('wisc_bc_data.csv')
```

In [38]:

```python
data.head()
```

Out[38]:

| | id | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | poin |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 87139402 | B | 12.32 | 12.39 | 78.85 | 464.1 | 0.10280 | 0.06981 | 0.03987 | |
| **1** | 8910251 | B | 10.60 | 18.95 | 69.28 | 346.4 | 0.09688 | 0.11470 | 0.06387 | |
| **2** | 905520 | B | 11.04 | 16.83 | 70.92 | 373.2 | 0.10770 | 0.07804 | 0.03046 | |
| **3** | 868871 | B | 11.28 | 13.39 | 73.00 | 384.8 | 0.11640 | 0.11360 | 0.04635 | |
| **4** | 9012568 | B | 15.19 | 13.21 | 97.65 | 711.8 | 0.07963 | 0.06934 | 0.03393 | |

5 rows × 32 columns

In [39]:

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 569 non-null    int64  
 1   diagnosis          569 non-null    object 
 2   radius_mean        569 non-null    float64
 3   texture_mean       569 non-null    float64
 4   perimeter_mean     569 non-null    float64
 5   area_mean          569 non-null    float64
 6   smoothness_mean    569 non-null    float64
 7   compactness_mean   569 non-null    float64
 8   concavity_mean     569 non-null    float64
 9   points_mean        569 non-null    float64
 10  symmetry_mean      569 non-null    float64
 11  dimension_mean     569 non-null    float64
 12  radius_se          569 non-null    float64
 13  texture_se         569 non-null    float64
 14  perimeter_se       569 non-null    float64
 15  area_se            569 non-null    float64
 16  smoothness_se      569 non-null    float64
 17  compactness_se     569 non-null    float64
 18  concavity_se       569 non-null    float64
 19  points_se          569 non-null    float64
 20  symmetry_se        569 non-null    float64
 21  dimension_se       569 non-null    float64
 22  radius_worst       569 non-null    float64
 23  texture_worst      569 non-null    float64
 24  perimeter_worst    569 non-null    float64
 25  area_worst         569 non-null    float64
 26  smoothness_worst   569 non-null    float64
 27  compactness_worst  569 non-null    float64
 28  concavity_worst    569 non-null    float64
 29  points_worst       569 non-null    float64
 30  symmetry_worst     569 non-null    float64
 31  dimension_worst    569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [40]:

```python
data.dtypes
```

Out[40]:

```
id                     int64
diagnosis             object
radius_mean          float64
```

```
texture_mean          float64
perimeter_mean        float64
area_mean             float64
smoothness_mean       float64
compactness_mean      float64
concavity_mean        float64
points_mean           float64
symmetry_mean         float64
dimension_mean        float64
radius_se             float64
texture_se            float64
perimeter_se          float64
area_se               float64
smoothness_se         float64
compactness_se        float64
concavity_se          float64
points_se             float64
symmetry_se           float64
dimension_se          float64
radius_worst          float64
texture_worst         float64
perimeter_worst       float64
area_worst            float64
smoothness_worst      float64
compactness_worst     float64
concavity_worst       float64
points_worst          float64
symmetry_worst        float64
dimension_worst       float64
dtype: object
```

In [41]:
```
data.shape
```

Out[41]:
```
(569, 32)
```

In [42]:
```
data.isnull().sum()
```

Out[42]:
```
id                    0
diagnosis             0
radius_mean           0
texture_mean          0
perimeter_mean        0
area_mean             0
smoothness_mean       0
compactness_mean      0
concavity_mean        0
points_mean           0
symmetry_mean         0
dimension_mean        0
radius_se             0
texture_se            0
perimeter_se          0
area_se               0
smoothness_se         0
compactness_se        0
concavity_se          0
points_se             0
symmetry_se           0
dimension_se          0
radius_worst          0
texture_worst         0
perimeter_worst       0
area_worst            0
smoothness_worst      0
compactness_worst     0
concavity_worst       0
points_worst          0
symmetry_worst        0
dimension_worst       0
dtype: int64
```

there are no missing values

In [43]:
```
data=pd.get_dummies(data,columns=['diagnosis'])
```

In [44]:
```
data.head()
```

Out[44]:

| | id | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | points_mean | s |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 87139402 | 12.32 | 12.39 | 78.85 | 464.1 | 0.10280 | 0.06981 | 0.03987 | 0.03700 | |
| **1** | 8910251 | 10.60 | 18.95 | 69.28 | 346.4 | 0.09688 | 0.11470 | 0.06387 | 0.02642 | |
| **2** | 905520 | 11.04 | 16.83 | 70.92 | 373.2 | 0.10770 | 0.07804 | 0.03046 | 0.02480 | |
| **3** | 868871 | 11.28 | 13.39 | 73.00 | 384.8 | 0.11640 | 0.11360 | 0.04635 | 0.04796 | |
| **4** | 9012568 | 15.19 | 13.21 | 97.65 | 711.8 | 0.07963 | 0.06934 | 0.03393 | 0.02657 | |

5 rows × 33 columns

In [45]:

```
data.median()
```

Out[45]:

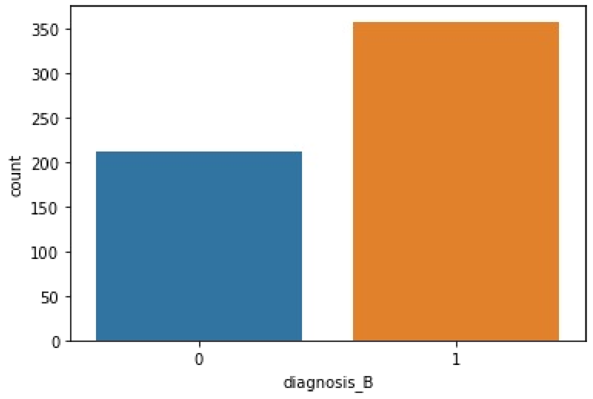
```
id                     906024.000000
radius_mean                13.370000
texture_mean               18.840000
perimeter_mean             86.240000
area_mean                 551.100000
smoothness_mean             0.095870
compactness_mean            0.092630
concavity_mean              0.061540
points_mean                 0.033500
symmetry_mean               0.179200
dimension_mean              0.061540
radius_se                   0.324200
texture_se                  1.108000
perimeter_se                2.287000
area_se                    24.530000
smoothness_se               0.006380
compactness_se              0.020450
concavity_se                0.025890
points_se                   0.010930
symmetry_se                 0.018730
dimension_se                0.003187
radius_worst               14.970000
texture_worst              25.410000
perimeter_worst            97.660000
area_worst                686.500000
smoothness_worst            0.131300
compactness_worst           0.211900
concavity_worst             0.226700
points_worst                0.099930
symmetry_worst              0.282200
dimension_worst             0.080040
diagnosis_B                 1.000000
diagnosis_M                 0.000000
dtype: float64
```

In [46]:

```
sns.countplot(x="diagnosis_B",data=data)
```

Out[46]:

```
<AxesSubplot:xlabel='diagnosis_B', ylabel='count'>
```



In [49]:

```
sns.countplot(x="diagnosis_M",hue="radius_worst",data=data)
```

Out[49]:

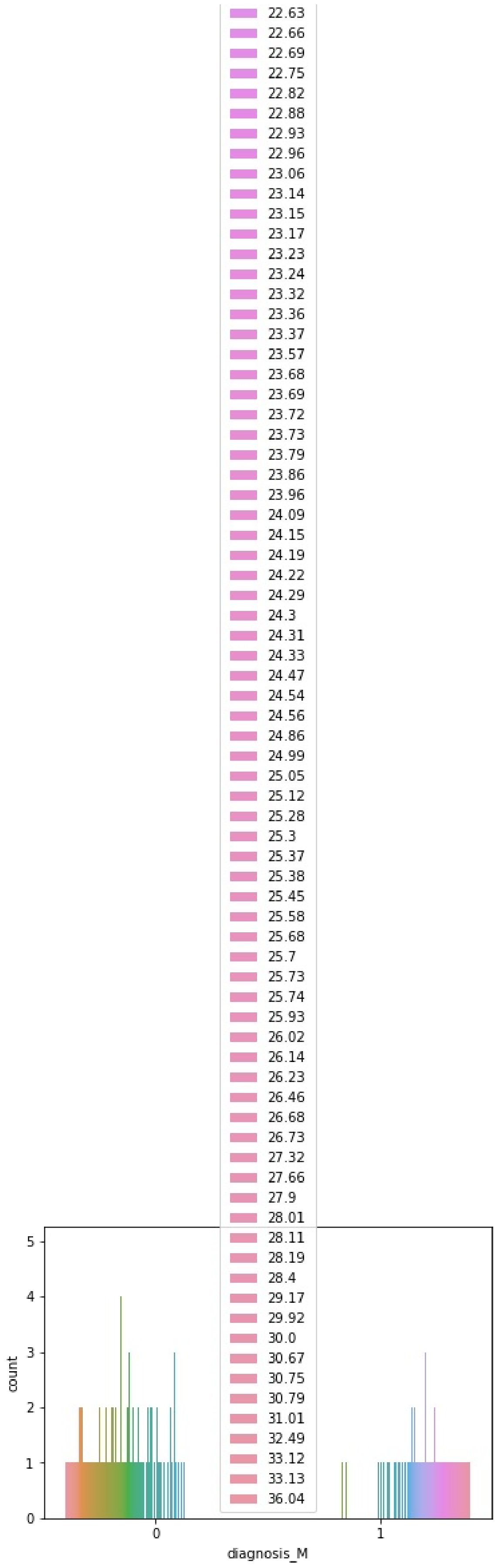
```
<AxesSubplot:xlabel='diagnosis_M', ylabel='count'>
```

| | |
|---|---|
| 🟥 | 8.952 |
| 🟥 | 8.964 |
| 🟥 | 9.077 |
| 🟥 | 9.092 |
| 🟥 | 9.262 |
| 🟥 | 9.414 |
| 🟥 | 9.456 |
| 🟥 | 9.473 |
| 🟥 | 9.507 |
| 🟥 | 9.565 |
| 🟥 | 9.628 |
| 🟥 | 9.699 |
| 🟥 | 9.733 |
| 🟥 | 9.845 |
| 🟥 | 9.965 |
| 🟥 | 9.968 |
| 🟥 | 9.981 |
| 🟥 | 10.01 |
| 🟥 | 10.06 |
| 🟥 | 10.17 |
| 🟥 | 10.23 |
| 🟥 | 10.28 |
| 🟥 | 10.31 |
| 🟧 | 10.41 |
| 🟧 | 10.42 |
| 🟧 | 10.49 |
| 🟧 | 10.51 |
| 🟧 | 10.57 |
| 🟧 | 10.6 |
| 🟧 | 10.62 |
| 🟧 | 10.65 |
| 🟧 | 10.67 |
| 🟧 | 10.75 |
| 🟧 | 10.76 |
| 🟧 | 10.83 |
| 🟧 | 10.84 |
| 🟧 | 10.85 |
| 🟧 | 10.88 |
| 🟧 | 10.92 |
| 🟧 | 10.93 |
| 🟧 | 10.94 |
| 🟧 | 11.02 |
| 🟧 | 11.05 |
| 🟧 | 11.06 |
| 🟧 | 11.11 |
| 🟧 | 11.14 |
| 🟧 | 11.15 |
| 🟧 | 11.16 |
| 🟧 | 11.17 |
| 🟧 | 11.21 |
| 🟧 | 11.24 |
| 🟧 | 11.25 |
| 🟧 | 11.26 |
| 🟧 | 11.28 |
| 🟧 | 11.35 |
| 🟧 | 11.37 |
| 🟧 | 11.38 |
| 🟨 | 11.48 |
| 🟨 | 11.52 |
| 🟨 | 11.54 |
| 🟨 | 11.6 |
| 🟨 | 11.62 |
| 🟨 | 11.66 |
| 🟨 | 11.68 |
| 🟨 | 11.69 |
| 🟨 | 11.86 |
| 🟨 | 11.87 |
| 🟨 | 11.88 |
| 🟨 | 11.92 |
| 🟨 | 11.93 |
| 🟨 | 11.94 |
| 🟨 | 11.95 |
| 🟨 | 11.98 |
| 🟨 | 11.99 |
| 🟨 | 12.02 |
| 🟨 | 12.04 |
| 🟨 | 12.08 |
| 🟨 | 12.09 |
| 🟨 | 12.12 |
| 🟨 | 12.13 |
| 🟨 | 12.2 |
| 🟨 | 12.25 |
| 🟨 | 12.26 |
| 🟨 | 12.32 |
| 🟨 | 12.33 |
| 🟨 | 12.34 |
| 🟨 | 12.36 |
| 🟨 | 12.37 |
| 🟨 | 12.4 |
| 🟨 | 12.41 |
| 🟨 | 12.44 |
| 🟨 | 12.45 |
| 🟨 | 12.47 |
| 🟨 | 12.48 |
| 🟨 | 12.51 |

🟩 12.57
🟩 12.58
🟩 12.61
🟩 12.64
🟩 12.65
🟩 12.68
🟩 12.76
🟩 12.77
🟩 12.78
🟩 12.79
🟩 12.81
🟩 12.82
🟩 12.83
🟩 12.84
🟩 12.85
🟩 12.88
🟩 12.9
🟩 12.97
🟩 12.98
🟩 13.01
🟩 13.03
🟩 13.05
🟩 13.06
🟩 13.07
🟩 13.09
🟩 13.1
🟩 13.11
🟩 13.12
🟩 13.13
🟩 13.14
🟩 13.15
🟩 13.16
🟩 13.18
🟩 13.19
🟩 13.2
🟩 13.24
🟩 13.25
🟩 13.28
🟩 13.29
🟩 13.3
🟩 13.31
🟩 13.32
🟩 13.33
🟩 13.34
🟩 13.35
🟩 13.36
🟩 13.37
🟩 13.45
🟩 13.46
🟩 13.5
🟩 13.56
🟩 13.57
🟩 13.58
🟩 13.59
🟩 13.6
🟩 13.61
🟩 13.62
🟩 13.63
🟩 13.64
🟩 13.65
🟩 13.67
🟩 13.71
🟩 13.72
🟩 13.74
🟩 13.75
🟩 13.76
🟩 13.78
🟩 13.8
🟩 13.82
🟩 13.83
🟩 13.86
🟩 13.87
🟩 13.88
🟩 13.89
🟩 13.9
🟩 13.94
🟩 14.0
🟩 14.04
🟩 14.06
🟩 14.08
🟩 14.09
🟩 14.1
🟩 14.11
🟩 14.13
🟩 14.16
🟩 14.17
🟩 14.18
🟩 14.19
🟩 14.2
🟩 14.23
🟩 14.24
🟩 14.26
🟩 14.29
🟩 14.34
🟩 14.35

🟩 14.37
🟩 14.38
🟩 14.39
🟩 14.4
🟩 14.41
🟩 14.42
🟩 14.44
🟩 14.45
🟩 14.48
🟩 14.49
🟩 14.5
🟩 14.54
🟩 14.55
🟩 14.62
🟩 14.67
🟩 14.69
🟩 14.73
🟩 14.77
🟩 14.8
🟩 14.83
🟩 14.84
🟩 14.85
🟩 14.9
🟩 14.91
🟩 14.92
🟩 14.96
🟩 14.97
🟩 14.98
🟩 14.99
🟩 15.01
🟩 15.03
🟩 15.05
🟩 15.09
🟩 15.1
🟩 15.11
🟩 15.14
🟩 15.15
🟩 15.2
🟩 15.27
🟩 15.29
🟩 15.3
🟩 15.33
🟩 15.34
🟩 15.35
🟩 15.4
🟩 15.44
🟩 15.47
🟩 15.48
🟩 15.49
🟩 15.5
🟩 15.51
🟩 15.53
🟩 15.61
🟩 15.63
🟩 15.65
🟩 15.66
🟩 15.67
🟩 15.7
🟩 15.74
🟩 15.75
🟩 15.77
🟩 15.79
🟩 15.8
🟩 15.85
🟩 15.89
🟩 15.93
🟩 15.98
🟩 16.01
🟩 16.08
🟩 16.11
🟩 16.2
🟩 16.21
🟩 16.22
🟩 16.23
🟩 16.25
🟩 16.3
🟩 16.31
🟩 16.33
🟩 16.34
🟩 16.35
🟩 16.36
🟩 16.39
🟩 16.41
🟩 16.43
🟩 16.45
🟩 16.46
🟩 16.51
🟩 16.57
🟩 16.67
🟩 16.76
🟩 16.77
🟩 16.82
🟩 16.84
🟩 16.86
🟩 16.89

▬ 16.97
▬ 16.99
▬ 17.01
▬ 17.04
▬ 17.06
▬ 17.09
▬ 17.11
▬ 17.18
▬ 17.26
▬ 17.27
▬ 17.31
▬ 17.32
▬ 17.36
▬ 17.38
▬ 17.39
▬ 17.46
▬ 17.5
▬ 17.52
▬ 17.58
▬ 17.62
▬ 17.67
▬ 17.71
▬ 17.73
▬ 17.77
▬ 17.79
▬ 17.8
▬ 17.87
▬ 17.91
▬ 17.98
▬ 18.07
▬ 18.1
▬ 18.13
▬ 18.22
▬ 18.23
▬ 18.33
▬ 18.49
▬ 18.51
▬ 18.55
▬ 18.76
▬ 18.79
▬ 18.81
▬ 18.98
▬ 19.07
▬ 19.18
▬ 19.19
▬ 19.2
▬ 19.26
▬ 19.28
▬ 19.38
▬ 19.47
▬ 19.56
▬ 19.59
▬ 19.76
▬ 19.77
▬ 19.8
▬ 19.82
▬ 19.85
▬ 19.92
▬ 19.96
▬ 20.01
▬ 20.05
▬ 20.11
▬ 20.19
▬ 20.21
▬ 20.27
▬ 20.33
▬ 20.38
▬ 20.39
▬ 20.42
▬ 20.47
▬ 20.58
▬ 20.6
▬ 20.8
▬ 20.82
▬ 20.88
▬ 20.92
▬ 20.96
▬ 20.99
▬ 21.08
▬ 21.2
▬ 21.31
▬ 21.44
▬ 21.53
▬ 21.57
▬ 21.58
▬ 21.65
▬ 21.84
▬ 21.86
▬ 22.03
▬ 22.25
▬ 22.32
▬ 22.39
▬ 22.51
▬ 22.52
▬ 22.54



In [48]:
```
data.describe()
```

Out[48]:

| | id | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | points |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 5.690000e+02 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.0 |
| **mean** | 3.037183e+07 | 14.127292 | 19.289649 | 91.969033 | 654.889104 | 0.096360 | 0.104341 | 0.088799 | 0.0 |
| **std** | 1.250206e+08 | 3.524049 | 4.301036 | 24.298981 | 351.914129 | 0.014064 | 0.052813 | 0.079720 | 0.0 |
| **min** | 8.670000e+03 | 6.981000 | 9.710000 | 43.790000 | 143.500000 | 0.052630 | 0.019380 | 0.000000 | 0.0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **25%** | 8.692180e+05 | 11.700000 | 16.170000 | 75.170000 | 420.300000 | 0.086370 | 0.064920 | 0.029560 | 0.0 |
| **50%** | 9.060240e+05 | 13.370000 | 18.840000 | 86.240000 | 551.100000 | 0.095870 | 0.092630 | 0.061540 | 0.0 |
| **75%** | 8.813129e+06 | 15.780000 | 21.800000 | 104.100000 | 782.700000 | 0.105300 | 0.130400 | 0.130700 | 0.0 |
| **max** | 9.113205e+08 | 28.110000 | 39.280000 | 188.500000 | 2501.000000 | 0.163400 | 0.345400 | 0.426800 | 0.2 |

8 rows × 33 columns

In [54]:
```
x=data.drop(["diagnosis_B","diagnosis_M"],axis=1)
y=data["diagnosis_B"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=41)
```

In [56]:
```
model=LogisticRegression()
model.fit(x_train,y_train)
```

Out[56]:
```
LogisticRegression()
```

In [57]:
```
pre=model.predict(x_test)
```

In [59]:
```
from sklearn.metrics import accuracy_score
```

In [60]:
```
accuracy_score(y_test,pre)
```

Out[60]:
```
0.5964912280701754
```

### decisiontree

In [67]:
```
dtree=DecisionTreeClassifier(criterion='gini',random_state=1)
dtree.fit(x_train,y_train)
```

Out[67]:
```
DecisionTreeClassifier(random_state=1)
```

In [68]:
```
print(dtree.score(x_train,y_train))
print(dtree.score(x_test,y_test))
```

```
1.0
0.9122807017543859
```

In [ ]:
```

```

### not a good model

In [65]:
```
dtree=DecisionTreeClassifier(criterion='entropy',random_state=1)
dtree.fit(x_train,y_train)
```

Out[65]:
```
DecisionTreeClassifier(criterion='entropy', random_state=1)
```

In [66]:
```
print(dtree.score(x_train,y_train))
print(dtree.score(x_test,y_test))
```

```
1.0
0.9005847953216374
```

## not a good model

In [74]:
```
dtree=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=1)
dtree.fit(x_train,y_train)
print(dtree.score(x_train,y_train))
print(dtree.score(x_test,y_test))
```

```
0.9773869346733668
0.9181286549707602
```

## this is performing well

In [77]:
```
dtree=DecisionTreeClassifier(criterion='entropy',max_depth=4,random_state=1)
dtree.fit(x_train,y_train)
print(dtree.score(x_train,y_train))
print(dtree.score(x_test,y_test))
```

```
0.9899497487437185
0.9122807017543859
```

In [85]:
```
bg=BaggingClassifier(n_estimators=50,base_estimator=dtree,random_state=41)
bg=bg.fit(x_train,y_train)
y_predict=bg.predict(x_test)
print(bg.score(x_test,y_test))
```

```
0.9707602339181286
```

In [90]:
```
ad=AdaBoostClassifier(n_estimators=50,random_state=41)
ad=ad.fit(x_train,y_train)
y_predict=ad.predict(x_test)
print(ad.score(x_test,y_test))
```

```
0.9824561403508771
```

## it is good score

In [88]:
```
gd=GradientBoostingClassifier(n_estimators=50,random_state=41)
gd=gd.fit(x_train,y_train)
y_predict=gd.predict(x_test)
print(gd.score(x_test,y_test))
```

```
0.9590643274853801
```

In [95]:
```
rf=RandomForestClassifier(n_estimators=50,random_state=41,max_features=10)
rf=rf.fit(x_train,y_train)
y_predict=rf.predict(x_test)
print(rf.score(x_test,y_test))
```

```
0.9883040935672515
```

## for this data RandomForestClassifier is the best Classifier

In [ ]:
```

```